University of Bahrain
College of Information Technology
Department of Computer Science
Semester I, 2014-2015
ITCS102/ITCS104/ITCS112 (Computer Programming II)

Final Exam

Date: 08th Jan 2015 Time: 11:30-13:30

STUDENT NAME	KEY
STUDENT ID	
SECTION #	

QUESTION #	PART #	MARKS		Comments
Q1	PART A	10		
	PART B	15		
	PART C	5		
Q2	PART A	8		
	PART B	4		
	PART C	4		
Q3	PART A	5		
	PART B	9		
TOTAL		**60**		

NOTE: THERE ARE (8) PAGES IN THIS TEST

WRITE ONLY **ONE** SOLUTION FOR EACH QUESTION

Question 1 [Pointers, Classes, & Arrays]

Consider the following class description to answer Part (A) & (B).

A class named ***Weather*** should be designed to hold the following information:

- *num*: an integer to represent the number of counteries.
- *cName*: a pointer of type string, representing one-dimensional array of size *num* to hold the names of the counteries. For example, Bahrain, Kawuit, Oman, ... etc.
- *temp*: a pointer to pointer of type integer, representing a dynamic two-dimensional array to hold the counteries' average tempratures in each month, i.e. *temp* is two-dimentional array of size *num X 12*.

An example is shown below for two counteries and their monthly tempratures (i.e *num* = 2).

cName	***temp***											
Bahrain	17	18	21	25	30	33	34	34	32	30	25	20
Kawuit	17	21	26	32	40	44	46	46	43	36	26	20

The class should also include the following member functions:

1. A **Constructor** with default value parameter for *num*=100. The function should create the dynamic for *cName* and *temp* of size *num* and *(num X 12*), respectively.
2. A **Copy Constructor**.
3. A **Destructor**.
4. A function named **readWeather.** The function should prompt the user to enter the countries' names and their monthly temperatures.
5. A function named **printWeather.** The function should display the countries' names and their monthly temperatures.
6. A function named **displayHighLowTemp.** The function should display the highest and lowest temperatures for each country. For example, for the countries given above, the function should display the following:

Country Name	***Highest Temperature***	***Lowest Temperature***
Bahrain	*34*	*17*
Kawuit	*46*	*17*

Question 1 [Pointers, Classes, and Arrays]

PART A [10 Points]

Declare the **Weather** class, don't include the functions implementation.

```
// Total = 10 pts
class Weather{
    private:
        string *cName; // 1.5 pt
        int **temp; //1.5 pt
        int num; // 1 pt

    public:
          // each 1 pt
        Weather(int = 30);
        ~Weather();
        Weather(const Weather&);
        void readData( );
        void outputData( );
        void displayTemp( );
};
```

Question 1 [Continue ..]

PART B [15 Points]:

Using the class defined in Part (A), write the implementation of the following functions:

1) **Constructor (7 points)**
2) **displayHighLowTemp (8 points)**

```
Weather :: Weather(int max){ // 1 pt
            num=max; // 1 pt
            cName = new string[num]; // 1 pt
            temp = new int * [num]; // 2 pt
            for(int i=0; i<num; i++) // 2 pts
             temp[i] = new int[12];

}
```

```
void Weather::displayHighLowTemp( ){
     cout<<"cName\t Highest \t Lowest \n";
     for(int r=0; r<num; r++){ // 1 pt
              cout<<cName[r]<<" ";

              // find max, min for each row
              int min=temp[r][0];  // 1 pt
              int max=temp[r][0];  // 1 pt

              for(int c=1; c<12; c++)  // 1 pt
                  {
                   if(temp[r][c]>max)  // 1.5 pt
                     max=temp[r][c];

                   if(temp[r][c] < min) // 1.5 pt
                     min=temp[r][c];
                  }
                   //output = 1 pt
              cout<< max<<"\t"<< min<<endl;
              }
}
```

Question 1 [Pointers , Classes, and Arrays]

PART C [5 points]: Find the output of the following program:

```
#include<iostream>
using namespace std;
class playClass {
private:
    int A;
    static int B;
public:
  void setA(int a){ A = a;}
  void print( ) { cout<<A<<" "<<B<<endl;}
  void compute( ){ B += A;  }
  playClass(int a=0){  A = a; }
};
int playClass::B=1;
int main( ) {
  playClass X(5);
  playClass Y;
  playClass * ptr = & X;
  X.print( );
  Y.print( );
  Y.setA(10);
  ptr->compute( );
  X.print( );
  Y.print( );
  (*ptr).print( );
  return 0;
  }
```

OUTPUT

// each line 1 pt = total = 5 pts

5 1
0 1
5 6
10 6
5 6

Question 2 [Overloading & Templates]

Consider the following class to answer the questions in Part (A), (B) & (C).

```
class myGrades{
 private:
    double grades[10]; // to represent student's grades
    int num; // to represent the actual number of  grades in the array

 public:
      myGrades( ); // a constructor to initialize num to zero

      void print( ); // a print function to output the grades

      void addGrade(double); // add  a new grade to the end of the array and increment num

      double findAverage( );   // return the average of the grades

       double findSum( ); // return the summation of the grades
};
```

PART A [8 Points]

Rewrite myGrades class as a template class to hold an array of any type.

```
// Total = 8 pts
template<class T> // 2 pt
class myGrades{
 private:
    T grades[10]; // 1 pts
    int num;

 public:
      myGrades(  );
      void print(  );
      void addGrade(T);  // 2 pts
      double findAverage(  );
      T findSum(  ); // 2 pts

    // 1 pt for not changing anything else
};
```

Question 2 [Continue ..]

PART B [4 Points]

Write the implementation of the function *findSum* defined in the template class in Part (A).

PART C [4 points]

Using the template class defined in Part (A), write a main function to create an object named *list* of type *myGrades* to hold an array of integers. Prompt the user to enter two grades and add them to *list*, and then compute and output the summation of the *list* grades using *findSum* function.

```
template<class T> // 1 pt
T myGrades<T>:: findSum( ){ // 1 pt

  T sum=0; // 1 pt
// compute and return sum = 1 pt
  for(int i=0; i<num; i++)
     sum+=grades[i];
  return sum;
}
```

```
int main(){
   myGrades <int> obj; // 2 pt
   int a, b;

   // reading 1 pt
   cout<<"Enter two grades:";
   cin>>a>>b;

   obj.addGrade(a);
   obj.addGrade(b);


   // 1 pt
   cout<<"sum= "<<obj.findSum()<<endl;

system("pause");
return 0;
}
```

Question 3 [Recursion]

PART A [5 Points]: Show the output of the following program:

```
void recFunc(char ch1, char ch2, int a)
{
if (a==0)
  cout<<endl;
else {
      cout<<ch1;
      recFunc (ch1, ch2, a-5);
      cout<<ch2;
     }
}
int main ( ) {
recFunc('#' , '@',10);
return 0;
}
```

OUTPUT

// each character 1 pt, new line 1 pt
// Total = 5 pts

##
@@

PART B [9 Points]: Consider the following struct:

```
struct studentType {
   long ID;     string name;     int age ;   };
```

Write a **recursive** function that takes as parameters an array of type *studentType* and the array size. The function should return the number of students with age less than 18. The function prototype is: *int countYoungStudents (studentType list[ ], int size);*

```
int CountYoungStudents (studentType list[ ], int size){
 if(size==0)  // Base case 2 pt
    return 0;
 else if(list[size-1].Age<18)  // 2 pts
   return 1 + CountYoungStudents (list, size-1);// 3 pts
 else
  return CountYoungStudents (list, size-1); // 2 pts

}
```